# Budget Formulation System (BFS)

# Test & Evaluation Master Plan


**Version 1.0**

**August 13, 2015**

## Contents

# 1 General Information

## 1.1 Purpose

The purpose of this document is to establish a software Test & Evaluation Master Plan (TEMP) for the Budget Formulation System (BFS) project. Stated most simply, test and evaluation is the mechanism through which requirements are verified and functionality is validated. Accordingly, test planning and execution are key elements of the requirements management, quality management, configuration management and stakeholder/communication management processes. This document is intended to be read within the context of the requirements traceability, quality management and configuration management documents created for the BFS project.

## 1.2 Scope

The TEMP is part of the Integrated Project Plan for BFS. This plan defines the overall strategy for ensuring the developed and implemented system conforms to all documented requirements. The TEMP describes the types of testing that will be acceptable for use at various points in the system's life cycle and what constitutes "successful" testing.

## 1.3 System Overview

This project, part of OCFO's approved system development sequencing plan, is intended to modernize the application supporting EPA's planning, budgeting and accountability processes. A full description of the need, purpose, scope, and development approach for this project is found in the "Budget Formulation System Concept of Operations" (BFS CONOPS) document.

## 1.4 References

The list below identifies key documents that underlie the Test and Evaluation activities carried out for the BFS project.

- EPA System Lifecycle Management Procedures
- EPA System Lifecycle Management Guidance
- Key Logic Project Management Guide
- Office of Budget Change Management Standard Operating Procedure - Budget Formulation System Concept of Operations,
- Budget Formulation System Quality Management Plan
- Budget Formulation System Requirements Management Plan
- Budget Formulation System Configuration Management Plan,

- Software Engineering Institute CMMI-Dev 1.3 - Validation and Verification

## 1.5 Acronyms and Abbreviations

| Abbreviation | Description |
|---|---|
| CI | Configuration Item |
| BFS | Budget Formulation System |
| CM | Configuration Management *alt* Configuration Manager |
| CMMI | Capability Maturity Model Integration |
| CMS | Configuration Management System |
| CONOPs | Concept of Operations |
| CR | Change Request |
| EPA | U.S. Environmental Protection Agency |
| GUI | Graphical User Interface |
| NIST | National Institute of Standards and Technology |
| OB | Office of Budget |
| OCFO | Office of the Chief Financial Officer |
| PAL | Project Asset Library |
| PM | Project Manager *alt* Project Management |
| PMP | Project Management Plan |
| QA | Quality Assurance |
| QM | Quality Management |
| RM | Requirements Management *alt* Requirements Manager |
| RTM | Requirements Traceability Matrix |
| RTP | Requirements Traceability Plan |
| SEI | Software Engineering Institute |
| SLCM | System Life Cycle Management |
| SME | Subject Matter Expert |
| TEMP | Test and Evaluation Master Plan |
| TFS | Team Foundation Services |

## 1.6 Roles and Responsibilities

System testing occurs throughout the system lifecycle. Effective testing involves the active participation of a variety of stakeholders, including software developers, business analysts, testing specialists, system owners and managers, business owners, end users, and IT infrastructure providers. The table below identifies the key roles and responsibilities involved in the BFS project.

| Role | Responsibility |
|---|---|
| Developer | Conduct unit tests; correct code defects and make code changes in response to test results |
| Test Team Member | Create use cases; carry out tests; report findings through TFS; verify technical requirements are met and validate against specified business requirements |
| Requirements Manager | Monitor tests; authorize changes to requirements as needed in response to test results; CI management involving requirements changes |
| Configuration Manager | Manages configuration changes resulting from tests |
| QA Manager | CI owner of test readiness and review report, test plans and cases, and defect records |
| Development Team Lead | Approves test plans; ensures tests conform with test plan including schedule; provides general oversight of test activities; CI owner of test plan documents |
| Project Manager | Authorizes changes to project plan as needed by test plan and test results; reviews / approves test reports; authorizes installation in production environment |
| EPA User Acceptance Team | Carry out user acceptance tests |

# 2. Software Testing and Quality Assurance

## 2.1 Test Management and Process

The test and evaluation framework is based on the project's planned development and deployment approach. (See Figure 1 below) As described more fully in the BFS Concept of Operations document, the project is using an iterative software development methodology in which requirements-based system features form the building blocks of each system build. As completed, associated features are grouped into capabilities which in turn are grouped into modules. Modules (see Section 2.2.2 below) form the basis of releases and support either a specific business process (e.g. budget formulation, strategic planning), system function (e.g., security, administration), or enhanced capability (e.g., modeling, analytics). Currently, four releases are planned between June 2015 and October 2016. Each release will include a combination of core business and enhanced capabilities.

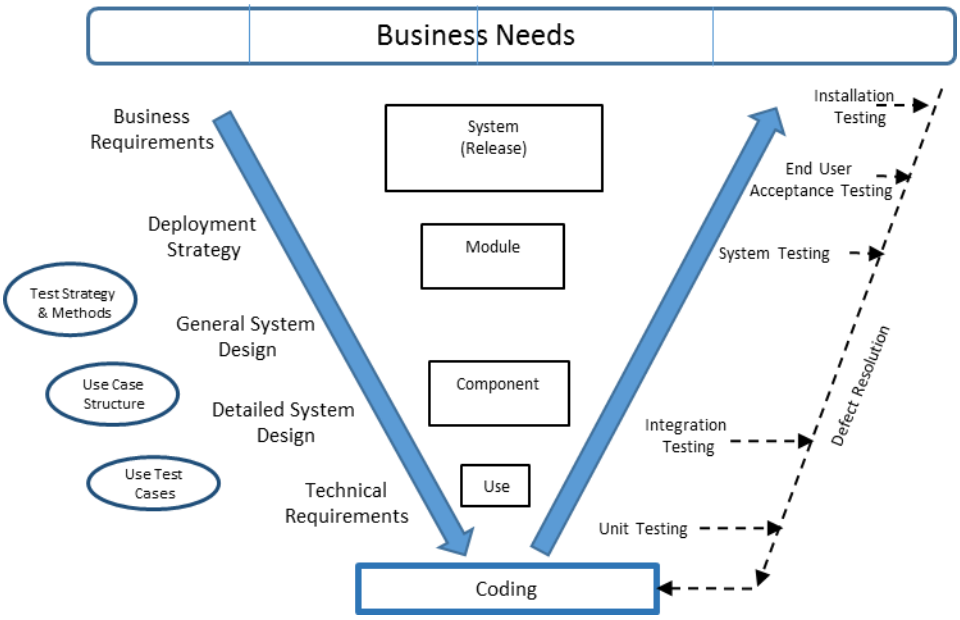



Figure 1

The tests themselves will be made part of the schedule and will be tested in accordance to scheduling needs. As discussed in the Requirements Traceability Plan (RTP) and in Section 3, below, the BFS project is using a set of automated tools to support software development, including testing. Figure 2, below, summarizes the interaction among tests and software builds leading to the deployment of a system release.

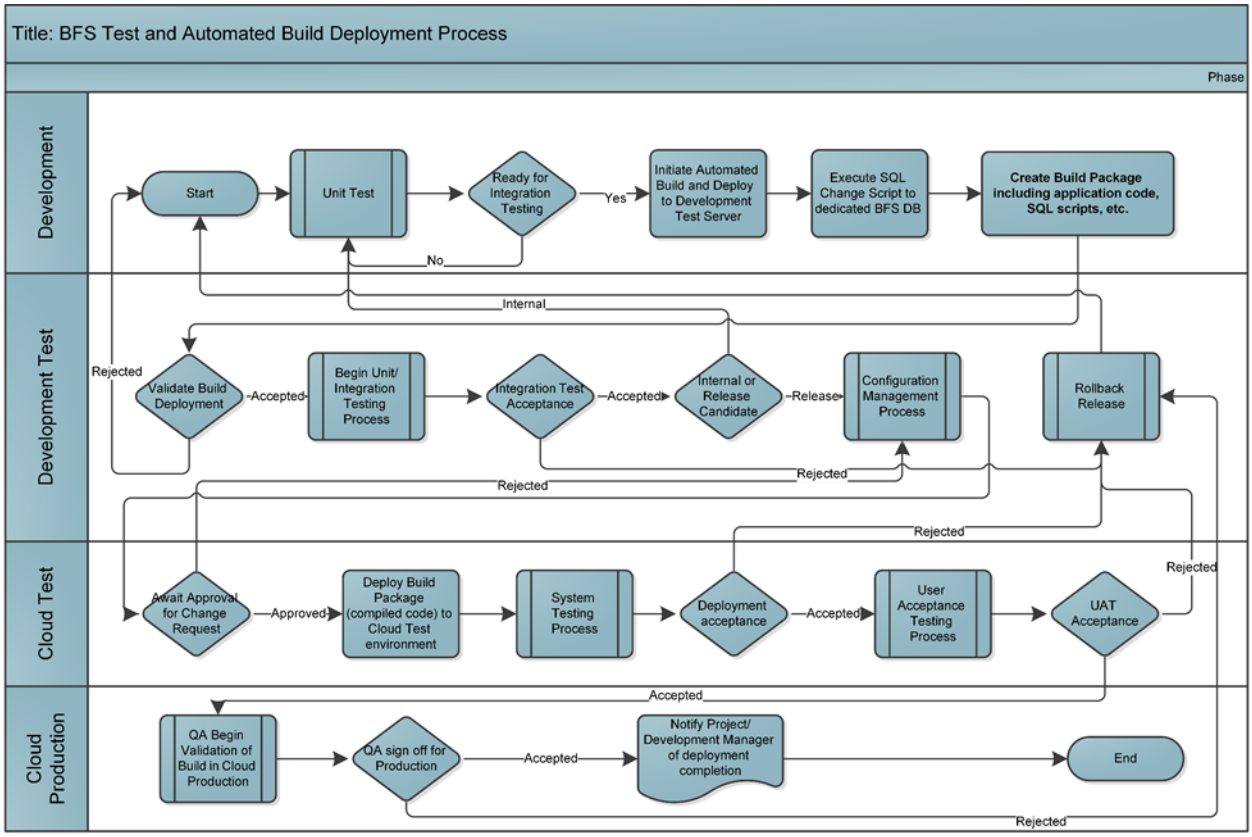


Figure 2

- The developer will complete development towards the Use Case.
- The developer will perform a Unit Test using the use case as the standard, inside of the development environment (which usually only has the developer's changes to code).
- If passed, the developer will change the state field in TFS from Active to Resolved and will assign the Use Case to the Integration Tester. The system will automatically assign, "Ready for Testing" in the Reason field.
- Once a build has been generated, the Integration Tester will test the changes in a test environment (which has all developer's changes for the latest build).
- If failed, the Integration Tester will change the State field inside of TFS to Active and assign the Use Case back to the developer. The system will automatically supply the reason as, "Test Failed." The integration tester will update the description field with the reproduction steps to cause the failure. Also, the integration tester will update the Use Case, annotating which step the test failed out, the Test Matrix, annotating the step that the test failed and the Known Issues list.
- If the test is passed, the integration test will change the Area Path of the Use Case to Drop N – Production Deployment Testing.
- Once the system is ready for Deployment, the Deployment Tester will test inside of the Deployment environment.
- If the Production Deployment Test is failed, the Deployment Tester will change the area path back to Drop N, and will reassign the State field back to Active and will reassign

ownership of the Use Case to the Developer. The system will automatically supply inside of the Reason field "Test Failed"

- If the Deployment Test is passed, the Deployment Tester will reassign the Area Path field of the Use Case to Drop N - System Testing.
- Once the system is ready for System Testing, the System Tester will test inside of the System environment.
- If the System Test is failed, the System Tester will change the area path back to Drop N, and will reassign the State field back to Active and will reassign ownership of the Use Case to the Developer. The system will automatically supply inside of the Reason field "Test Failed"
- If the System Test is passed, the System Tester will reassign the Area Path field of the Use Case to Drop N - User Acceptance Testing.
- Once the system is ready for User Acceptance Testing, the User Acceptance Tester will test inside of the User Acceptance Test environment.
- If the User Acceptance Test is failed, the User Acceptance Tester will change the area path back to Drop N, and will reassign the State field back to Active and will reassign ownership of the Use Case to the Developer. The system will automatically supply inside of the Reason field "Test Failed"
- If the User Acceptance Test is passed, the User Acceptance Tester will reassign the state field to, "Closed." The TFS system will automatically provide inside of the reason field "Test Passed."

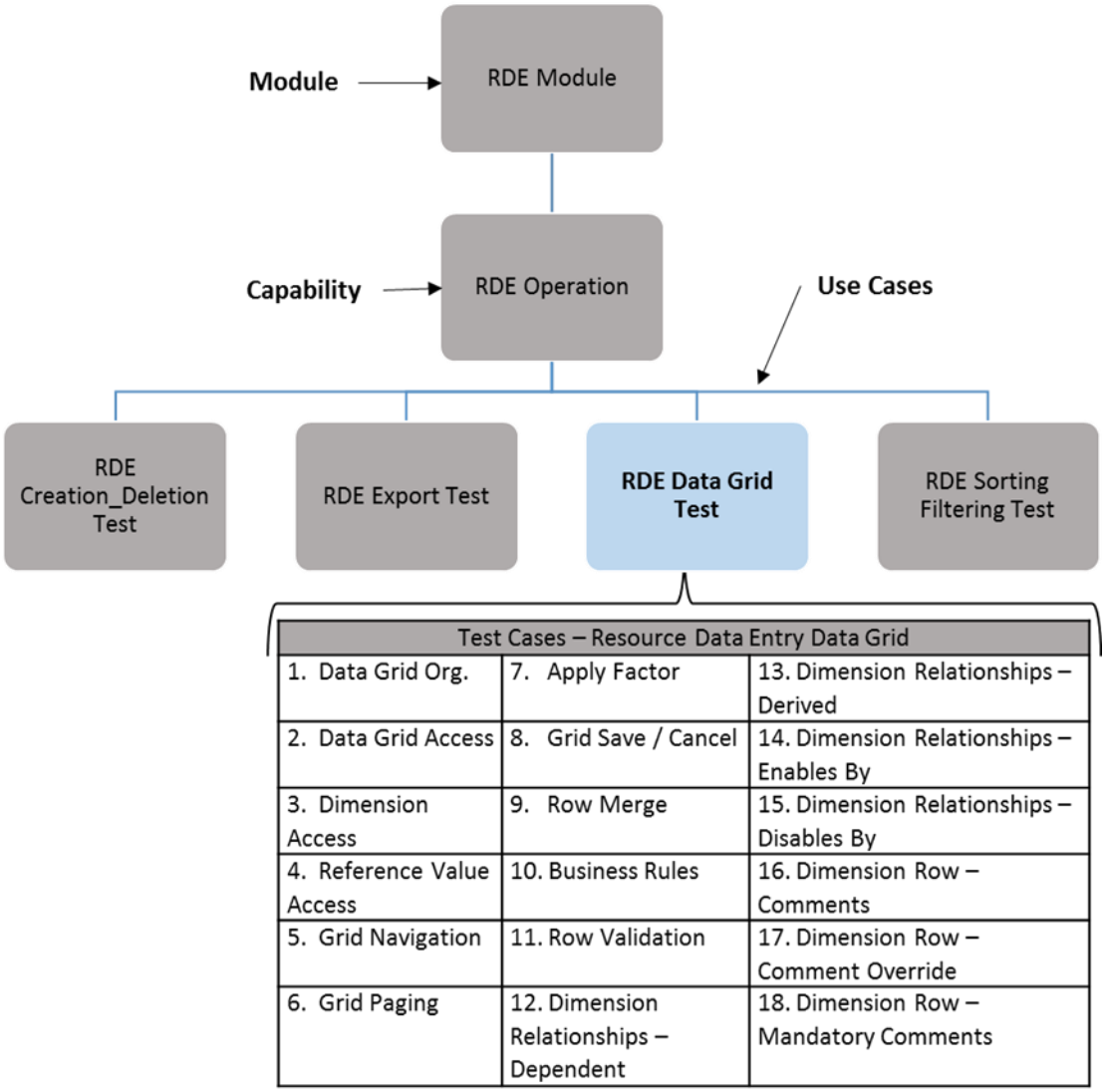
## 2.2 Test Phase Descriptions

Testing will consist of five phases: 1) Unit, 2) Integration, 3) System, 4) End User Acceptance, and 5) Installation. Due to the iterative nature of the development approach, elements of a release will be in various testing phases up to the point of release. Throughout all five phases, regression testing will be performed any time changes are made to an established baseline to ensure that those changes do not have unintended consequences.

### 2.2.1 Unit Testing

Unit testing is best described as the confirmation that the software unit in question does what the design document and use case says it should do. The Requirements Traceability Plan (RTP) describes the 10 step process of organizing requirements into feature based groups, developing code, preparing test cases to test the code against requirements, executing the tests and managing the test results.

Unit testing is based on test cases (see Figure 3 below) and is performed by the developers in their individual test environments prior to the code in question being compiled or added to an existing build.




| Test Cases – Resource Data Entry Data Grid | | |
|---|---|---|
| 1. Data Grid Org. | 7. Apply Factor | 13. Dimension Relationships – Derived |
| 2. Data Grid Access | 8. Grid Save / Cancel | 14. Dimension Relationships – Enables By |
| 3. Dimension Access | 9. Row Merge | 15. Dimension Relationships – Disables By |
| 4. Reference Value Access | 10. Business Rules | 16. Dimension Row – Comments |
| 5. Grid Navigation | 11. Row Validation | 17. Dimension Row – Comment Override |
| 6. Grid Paging | 12. Dimension Relationships – Dependent | 18. Dimension Row – Mandatory Comments |

Figure 3

The test cases and results including all errors and discrepancies that are found during the Unit testing phase are documented in the Project Asset Library (PAL) and Team Foundation Services (TFS), the project's application lifecycle management tool. If the results are not compliant with expected test results the issue will be resolved through established requirements, issues, and configuration management processes.

### 2.2.2 Integration Testing

Integration testing in the BFS project means component integration testing. Integration tests are tests which are conducted by the test team against multiple software units to verify and validate associated use cases. Integration tests involve partially complete releases; the scope varies from individual use cases through features and capabilities to complete modules (see figure above). Testing is initially done by the responsible code developer, then by a member of the project's test team. The following specific areas of integration testing include:

- *Screen Navigation*: Does the application perform as designed when moving from one area to another? Do all the controls work?
- *Data Manipulation*: When data is added, edited or deleted from the application, do the changes get correctly applied to the database?
- *Data Retrieval*: Is the correct data retrieved from queries and screen selections?
- *Data Accuracy*: Do reference tables contain all the information they are supposed to contain? Is data correctly transferred and stored from other systems?
- *Error Handling*: Do any unexpected errors occur? If so, are they handled correctly?

Integration testing focuses on how individual uses within a module inter-operate and how individual modules work with other modules scheduled to be included in the planned release. Appendix A identifies the modules and specific use cases to be tested as part of integration testing for the first release. Additional use cases will be developed as work on detailed designs of future releases are completed (see Figure 1). The table below summarizes the modules to be developed over the life of the project, and identifies those to be included in the initial release.

**BFS Modules**

| | | | |
|---|---|---|---|
| X | Alerts and Notifications | X | Application Security and Admin. |
| | Budget Document Production | X | Budget Versioning |
| X | Bulk Data Operations | X | Crosswalk/Transformation |
| | Distribution Accounts | | Cost Allocations |
| | Payroll Projection | X | Modeling |
| | Performance | | Flexible Data Structure |
| X | Predefined Reports / Queries | X | Reports and Queries |
| X | Reference Data Maintenance | | Reprogramming |
| X | Resource Data Entry / Import / Export | X | Strategic Planning |
| | System Interfaces | | |

X = Included in Release 1

### 2.2.3 System Testing

System testing, also known as end-to-end testing, begins with tests to ensure that the system release as a whole operates as designed. Additional tests of functionality will be carried out to ensure that the system meets the expected behaviors of business users and business requirements associated with the release.

System testing will be performed in an environment that mirrors as closely as possible the hardware and software environment to be used in production. In addition to functionality, system testing will address the following areas:

| Type | Purpose |
|---|---|
| Security | Verify that system meets security requirements and applicable security standards. |
| Performance | Determine responsiveness and stability of system under production conditions; measured against specified requirements. |
| Disaster Recovery | Determine ability of system to return to an operational state after an event causes loss of system use for a period of time sufficient to affect business activities. |
| Restoration | Determine the ability of the system to be returned to a specified state following a hardware failure, software crash or similar problem. |
| Interface | Determine the ability of the system to interoperate with external systems as designed in the solutions interface documents. |

Any defects or other issues uncovered through system testing will be tracked in TFS and will be managed through established requirements, issue and configuration management processes.

### 2.2.4 EPA End User Acceptance Testing

End User acceptance testing is performed by the end users of a system prior to system implementation. User acceptance testing focuses on the system's ability to meet its intended purpose. Emphasis during this phase of testing is on such things as the acceptability and ease of use for a specific task or set of tasks.

User acceptance testing for BFS will involve individuals representing the user groups most affected by the functionality of the release. These users will be given a variety of practical test cases on the system. These cases will be drawn from the functionality tests created for system testing. Any defects or other issues uncovered through user acceptance testing will be tracked in TFS and will be managed through established requirements, issues, and configuration management processes.

User acceptance testing will be complete when the system has been approved for use by the system owner. This approval may be conditional and may include the evaluation or implementation of changes managed through the BFS Change Management process.

### 2.2.5 Installation Testing

Installation testing verifies that the system or system change was successfully moved into production without error and that system performance was not degraded.

The depth of verification testing to be done is closely associated with the nature of the change being implemented (see the project's Configuration Management Plan). For planned releases and major changes, verification tests will address the following areas:

- *Installation*: Confirm execution of installation scripts, numbers and sizes of files transferred, compatibility with hosted environment
- *Functionality*: Repeat set of tests identified during system testing to confirm system behaves as expected
- *Performance*: Repeat set of tests identified during system testing to confirm system performs as benchmarked during system tests
- *Security*: Verify that the security requirements and standards are not negatively impacted by the change

## 2.3 Regression testing

Regression tests will be performed whenever changes are made to an established software baseline. Regression testing is conducted in parallel with other tests; its purpose is to confirm that unmodified code is not affected by the change. Regression testing is selective; a regression test specific to a defect will be created for any defect identified as more severe than low. A regression test will be complete when the intended result(s) specified in the test case is met as determined by the involved developer and test team member.

## 2.4 Security Testing

BFS has been determined to meet the criteria for a Moderate system security classification. Federal security standards for information systems are published in the National Institute of Standards and Technology's 800-53 publications. The current version of this document (Revision 4) identifies approximately 80 system or technology based controls.

A significant portion of the security requirements associated with these controls will be the responsibility of the system's cloud-based, FedRAMP certified Host provider (see the BFS Concept of Operations). Accordingly, this test plan will not include the controls that are the responsibility of the Cloud provider.

Also, this project will make use of 3rd party testing to meet EPA's SLCM requirements for security and accreditation. Responsibility for security testing, evaluation and reporting during the system and verification phases of testing will be the responsibility of that 3rd party. The remediation responsibility remains with BFS.

BFS includes a security module which is designed to meet the system's security requirements, including requirements associated with applicable NIST 800-53 controls.
Figure 4, below, identifies how unit and integration security testing will be organized.

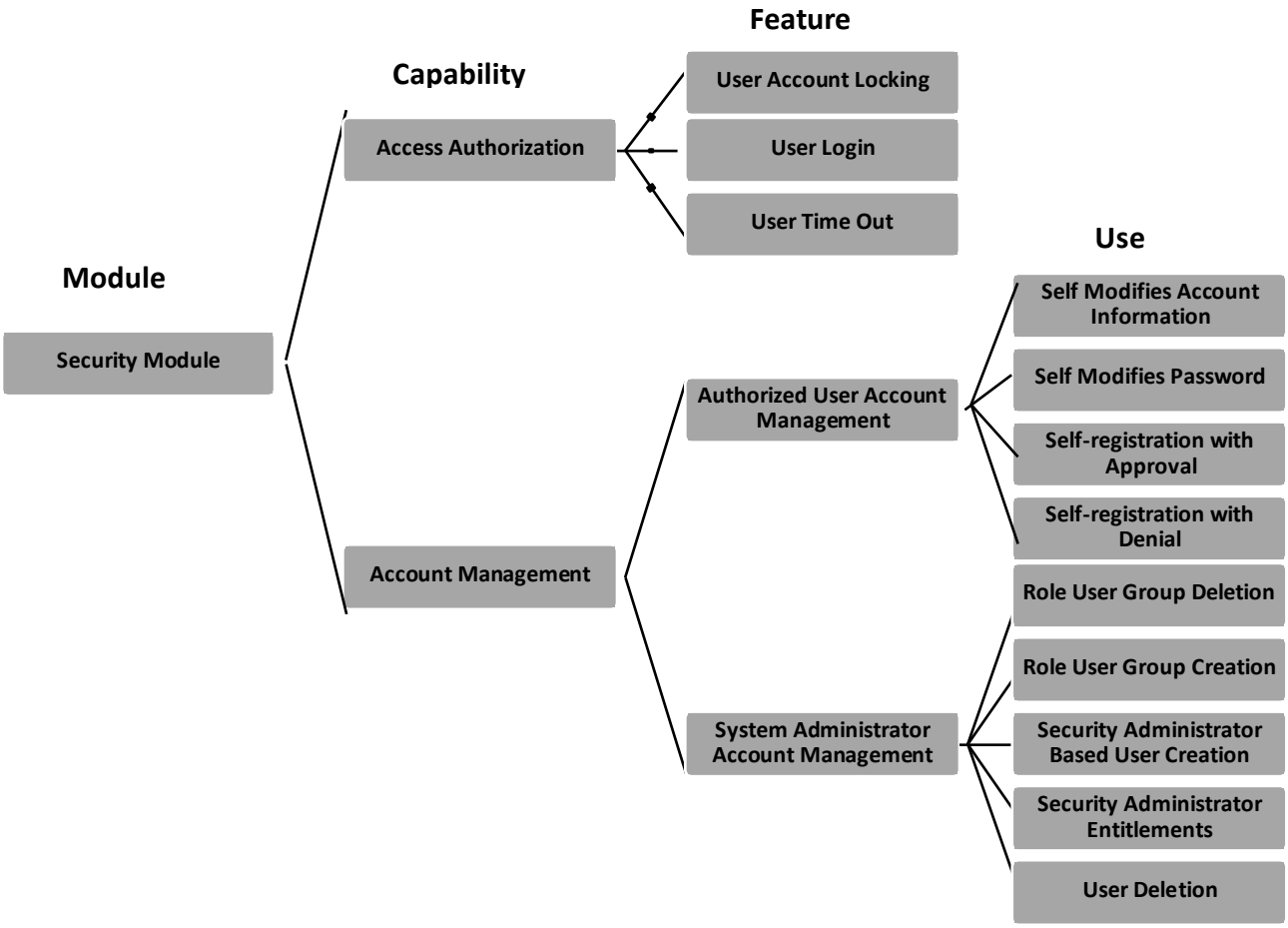



Figure 4

Security testing will occur in all test phases and will complete with the preparation of each release's Security and Accreditation package for the system's Authorization to Operate.

## 3. Test Tools and Infrastructure

As described in the Requirements Traceability Plan (RTP), Team Foundation Services and a SharePoint portal are the project's primary application lifecycle management tools. This includes support for managing and communicating tests and test results among the project developers, test team and other stakeholders. The RTP contains a sample of the Requirements Traceability Matrix, which includes information about the impacts of tests on requirements.

# 4. Test Reports

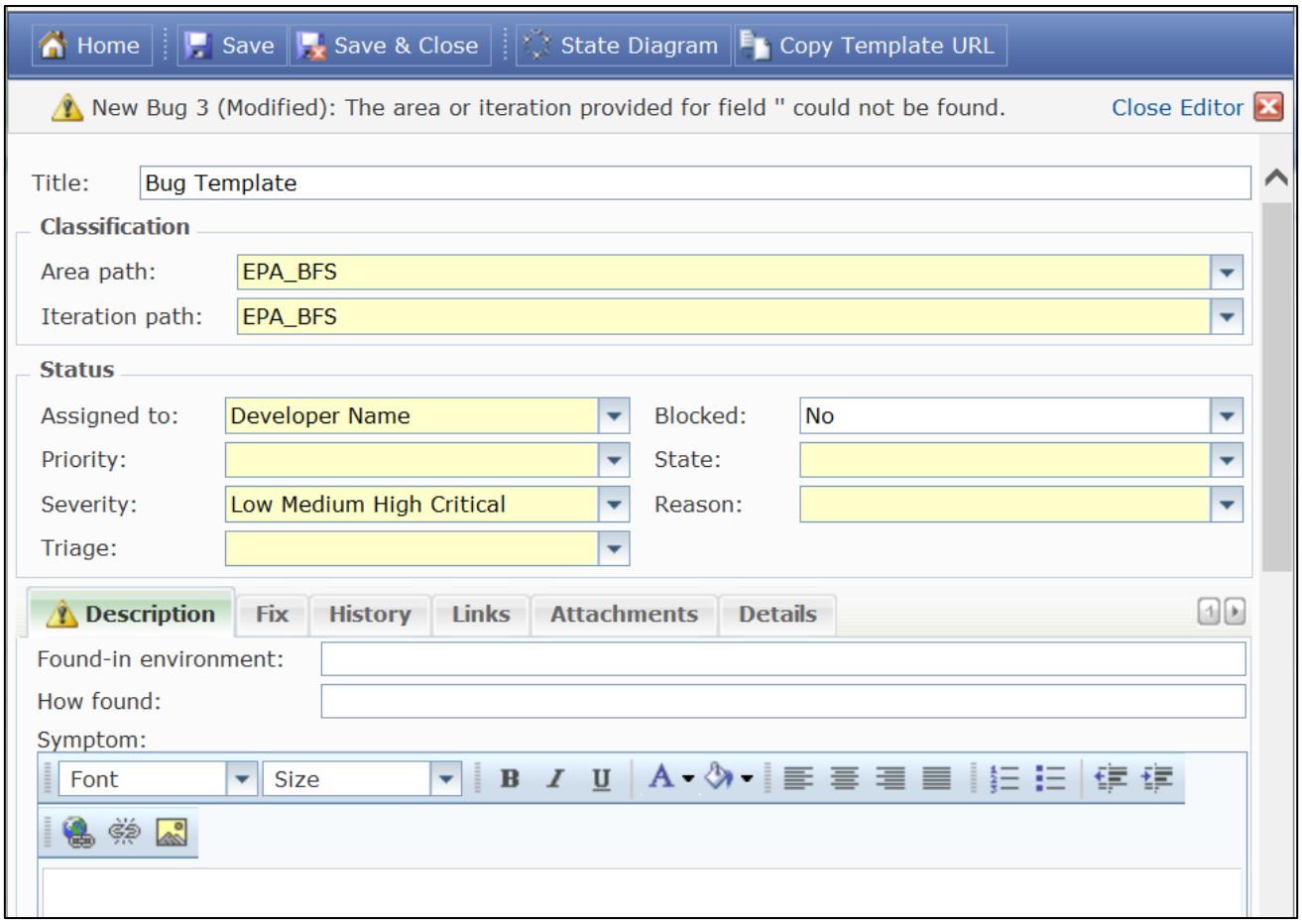
TFS provides a variety of reports, specialized for the various test roles. This includes detailed reports that capture the specific tests and test results to summary reports for senior management. Figure 5, below, is a template showing the information captured during resolution of a test issue.

Proposed Fix:

History:

Links:

Attachments:



Details:

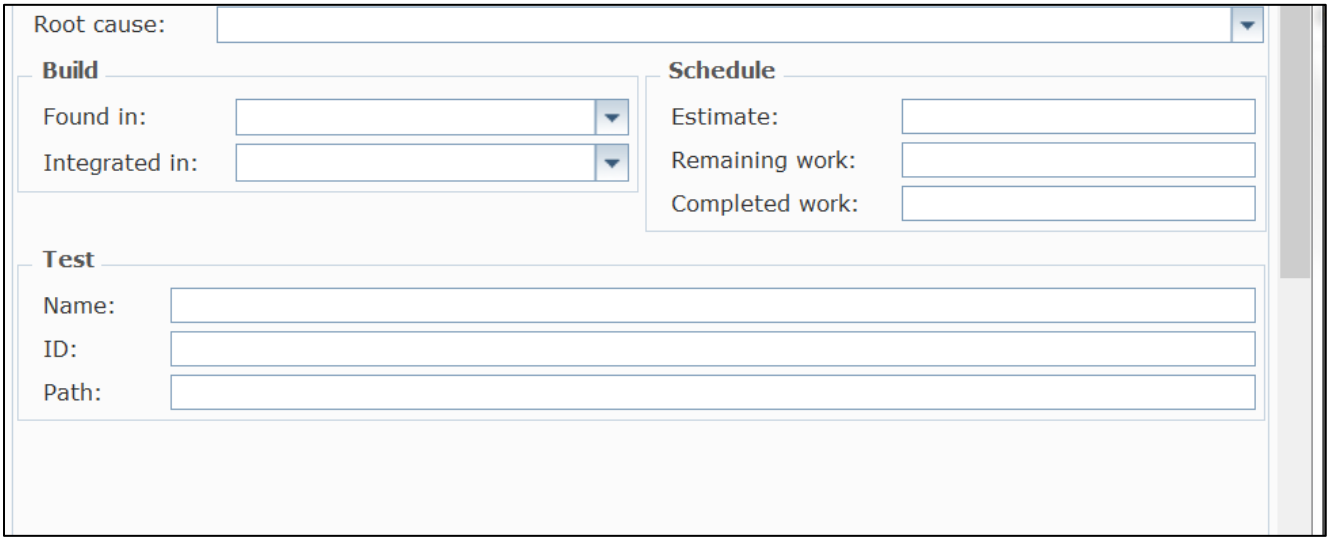



Figure 5

# Appendix A

BFS Use Test Cases by Module – Release 1

| Module Name / Use Test Case Title | |
|---|---|
| **Bulk Data Operations** | |
| | Bulk Data Operations: Copy |
| | Bulk Data Operations: Delete |
| | Bulk Data Operations: Merge |
| | Bulk Data Operations: Reset Resources |
| | Bulk Data Operations: Apply Factor |
| | Bulk Data Operations: Round Resources |
| | Bulk Data Operations: Transformation |
| | Bulk Data Operations: Transform |
| **Ad-hoc Report** | |
| | Ad-hoc Report GUI Capabilities |
| **Alerts and Notifications** | |
| | Alert Creation /_Deletion |
| | Alert GUI Functionality Test |
| **Budget Versioning** | |
| | Budget Version Status Screen GUI |
| | Create Budget Version |
| | Delete Budget Version |
| | Edit Budget Version |
| **Crosswalk** | |
| | Crosswalk GUI |
| | Crosswalk Operations |

| | |
|---|---|
| **Modeling** | |
| | Base Cost GUI |
| | Base Cost Operations |
| | Payroll Data and Import Logic GUI |
| | Payroll Data and Import Logic Operations |
| | Resource Specification GUI |
| | Resource Specification Operation |
| | Scenario GUI |
| | Scenario Operations |
| **Predefined Reports** | |
| | Predefined Report GUI |
| | Predefined Report Template |
| **Reference Data Maintenance** | |
| | Add Reference Table Values |
| | Delete Reference Table Values |
| | Edit Reference Dimension Description |
| | Edit Reference Table Values |
| | Reference Value GUI |
| | Import and Export Reference Table Values |
| **Resource Data Entry Import Export** | |
| | Creation Deletion Test |
| | RDE Import and Export |
| | RDE Data Grid Test |
| | Sorting Filtering |
| **Security** | |
| | Security Administrator Application Entitlements |

| | |
|---|---|
| | Role_ User Group Deletion |
| | Role_ User Group Creation |
| | Security Administrator Based user Creation |
| | User Account Locking |
| | User Deletion |
| | User Login |
| | User Password Modification |
| | User Self Registration |
| **Strategic Plan** | |
| | Strategic Plan: Creation |
| | Strategic Plan: Deletion |
| | Strategic Plan: Element Creation |
| | Strategic Plan: Element Deletion |
| | Strategic Plan: Element Modification |
| | Strategic Plan: GUI |
| | Strategic Plan: Modification |
| | Strategic Plan: Import and Export |
| | Strategic Plan: Structure Management Creation |
| | Strategic Plan: Structure Management Deletion |